# 9.5 例子

在下面的几个小节中，我们将介绍一些用于实现特殊效果的混合系数组合值。在这些例子中，我们主要关注于RGB混合，alpha混合的处理是类似的。

## 9.5.1 屏蔽颜色写入

假设我们希望原始目标像素保持不变，即不被任何其他数值覆盖，也不与当前的光栅化源像素进行混合。当我们希望屏蔽后台缓冲区、只向深度/模板缓冲区写入数据时，该算法非常有用。要实现这一算法，可将源像素混合系数设为**D3D11\_BLEND\_ZERO**，目标混合系数设为**D3D11\_BLEND\_ONE**，混合运算符设为**D3D11\_BLEND\_OP\_ADD**。使用一方案，混合方程可简化为：

**C** = **C**src ⊗ **F**src ⊞ **C**dst ⊗ **F**dst

**C** = **C**src ⊗ (0, 0,0) + **C**dst ⊗ (1, 1,1)

**C** = **C**dst

还有一种方法可以实现相同的结果，就是将**D3D11\_RENDER\_TARGET\_BLEND\_DESC::RenderTargetWriteMask**设置为0，这样就不会写入任何颜色通道。

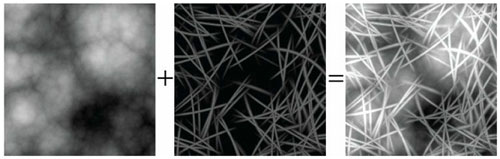
## 9.5.2 加法和减法

假设我们希望将源像素和目标像素相加（参见图9.2）。要实现这一算法，可将源混合系数设为**D3D11\_BLEND\_ONE**，目标混合系数设为**D3D11\_BLEND\_ONE**，混合运算符设为**D3D11\_BLEND\_OP\_ADD**。使用一方案，混合方程可简化为：

**C** = **C**src ⊗ **F**src ⊞ **C**dst ⊗ **F**dst

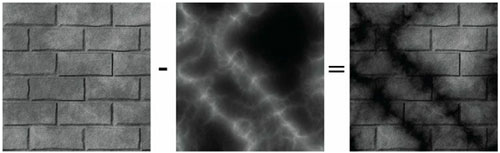
**C** = **C**src ⊗ (1, 1,1) + **C**dst ⊗ (1, 1,1)

**C** = **C**src + **C**dst

****

**图9.2 将源颜色和目标颜色相加。由于颜色叠加，所以创建后的图像比原始图像的亮度要高一些。**

当把**D3D11\_BLEND\_OP\_ADD**替换为**D3D11\_BLEND\_OP\_SUBTRACT**或**D3D11\_BLEND\_OP\_REV\_SUBTRACT**时，可以实现源像素和目标像素的减法运算（参见图9.3）。

****

**图9.3 从目标颜色中减去源颜色。由于颜色削减，所以创建后的图像比原始图像的亮度要低一些。**

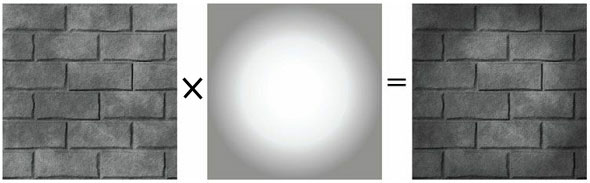
## 9.5.3 乘法

假设我们要将源像素和目标像素相乘（参见图9.4）。要实现这一算法，可将源混合系数设为**D3D11\_BLEND\_ZERO**，目标混合系数设为**D3D11\_BLEND\_SRC\_COLOR**，混合运算符设为**D3D11\_BLEND\_OP\_ADD**。使用一方案，混合方程可简化为：

**C** = **C**src ⊗ **F**src ⊞ **C**dst ⊗ **F**dst

**C** = **C**src ⊗ (0, 0,0) + **C**dst ⊗ **C**src

**C** = **C**dst⊗ **C**src

****

**图9.4 将源颜色和目标颜色相乘。**

## 9.5.4 透明度

我们可以使用源alpha分量*a*s控制源像素的不透明度（例如，当alpha为0.0时表示0%不透明，为0.4时表示40%不透明，为1.0时表示100%不透明）。不透明和透明之间的关系可以简单地表示为*T* =1 − *A*，其中*A*表示不透明度，*T*表示透明度。例如，当某物的不透明度为0.4时，它的透明度为1 − 0.4 = 0.6。现在，我们希望根据源像素的不透明度来混合源像素和目标像素。要实现一算法，可将源混合系数设为**D3D11\_BLEND\_SRC\_ALPHA**，目标混合系数设为D3D11\_BLEND\_INV\_SRC\_ALPHA，混合运算符设为**D3D11\_BLEND\_OP\_ADD**。使用这一方案，混合方程可简化为：

**C** = **C**src ⊗ **F**src ⊞ **C**dst ⊗ **F**dst

**C** = **C**src ⊗ (*a*s, *a*s, *a*s) + **C**dst ⊗ (1 - *a*s, 1 - *a*s,1 - *a*s)

**C** = *a*s**C**src +(1 - *a*s)**C**dst

例如，当*a*s = 0.25时，表示源像素的不透明度为25%， 或者说源像素的透明度为75%。那么，当源像素和目标像素混合之后，我们最终得到的颜色应该是由25%的源像素和75%的目标像素组成（源像素将“挡住”一部分目标像素）。综上所述，此时的混合方程为：

**C** = *a*s**C**src +(1 - *a*s)**C**dst

**C** = 0.25**C**dst + 0.75**C**src

通过一混合方式，我们可以绘制如图9.1所示的透明物体。但需要注意的是，此时物体的绘制顺序非常重要。我们需要遵守如下规则：

*首先绘制非透明物体。然后，根据透明物体与摄像机之间的距离进行排序，按照从后向前的顺序绘制透明物体。*

之所以要按照从后向前的顺序进行绘制，是为了让前面的物体和后面的物体进行混合。如果一个物体是透明的，那么我们就会透过这个物体看到它后面的其他物体。所以，必须将透明物体后面的所有物体先绘制出来，然后才能将透明的源像素和后台缓冲区中的目标像素进行混合。

对于9.5.1节的混合方程来说，绘制顺序并不重要，因为它只是简单地阻止源像素向后台缓冲区的写入操作。对于9.5.2和9.5.3节的混合方程，我们必须先绘制非透明物体，再绘制透明物体；因为我们需要在混合之前将所有的非透明几何体存入后台缓冲区。不过，我们不需要对透明物体进行排序，因为这些运算满足交换律。也就是，从后台缓冲区中的某个像素颜色**B**开始执行*n*次加/减/乘法运算，最终得到的混合颜色是一样的，与顺序无关：

**B**ʹ = **B** +**C**0 +**C**1 + ⋯ +**C**n-1

**B**ʹ = **B** - **C**0 - **C**1 - ⋯ - **C**n-1

**B**ʹ = **B** ⊗ **C**0 ⊗ **C**1 ⊗ ⋯⊗**C**n-1

## 9.5.5 混合与深度缓冲

当使用加/减/乘法混合时会出现一个与深度测试相关的问题。这里，我们仅以加法混合为例进行说明，同样的情况也适用于减法和乘法混合。当我们使用加法混合来渲染一个粒子系统*S*时，每个粒子是否相互遮挡并不重要；我们只需要把粒子的颜色简单地累加起来 （参见图9.5）。我们并不希望对*S*中的每个粒子进行深度测试。在这一情景中，如果不按照从后向前的顺序绘图，那么当*S*中的某个粒子被另一个粒子遮挡时，该粒子将无法通过深度测试，它的像素片段将被丢弃，也就是说该粒子的像素颜色不会累加到最终的混合颜色中。我们应该在渲染*S*时禁用深度写入功能，使粒子的深度信息不写入深度缓冲区。在禁用深度写入功能之后，由加法混合生成的粒子深度信息不会写入到深度缓冲区；所以，当*S*中的一个粒子被其他粒子遮挡时，该粒子依然可以通过深度测试并绘制到后台缓冲区中。注意，我们只在绘制*S*时禁用深度写入功能（以便于使用加法混合绘制该粒子系统），而深度读取和深度测试功能依然有效。非透明物体（在绘制透明物体之前）仍然可以遮挡位于它后面的透明物体。例如，当你在一堵墙的后面绘制一个透明物体时，该物体不会显示出来，因为它被墙体挡住了。我们会在下一章讲解有关深度写入和深度测试的设置方法。

****

**图9.5 当使用加法混合时，火焰中心的亮度比火焰边缘的亮度高，因为中间的粒子较多，它们彼此重叠并累加在一起。随着粒子逐渐散开，亮度会逐渐衰减，因为当重叠在一起的粒子越来越少时，累加值会相应降低。**